

Open Research Online

The Open University's repository of research publications
and other research outputs

Autonomous matchmaking web services

Conference or Workshop Item

How to cite:

Yu, Hong Qing; Dietze, Stefan and Benn, Neil (2010). Autonomous matchmaking web services. In: 2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM), pp. 420–425.

For guidance on citations see [FAQs](#).

© 2010 IEEE

Version: Version of Record

Link(s) to article on publisher's website:

<http://dx.doi.org/doi:10.1109/CISIM.2010.5643504>

Copyright and Moral Rights for the articles on this site are retained by the individual authors and/or other copyright owners. For more information on Open Research Online's data [policy](#) on reuse of materials please consult the policies page.

oro.open.ac.uk

Autonomous Matchmaking Web Services

Hong Qing Yu, Stefan Dietze and Neil Benn

Knowledge Media Institute
The Open University
Milton Keynes, United Kingdom
{h.q.yu, s.dietze, n.j.l.benn@open.ac.uk}

Abstract—Current Semantic Web Services research investigates how to dynamically discover assemble and invoke Web services. Despite many research efforts, Semantic Web Services are still not fully recognized in industry. One important reason is the dissevered description layers of syntax and semantics. In other words, semantics is only useful for a service broker to discover services whereas service requesters still need to invoke services based on syntactic descriptions. In this paper, we view semantics from another angle to reform the Web service framework completely (even for input messages and output messages during invocation) by using only RDF and Linked Open Data. We introduce Autonomous Matchmaking Web Services in which Web services are brokering themselves to notify the service registry whether they are suitable to the requesters. This framework is designated to more efficiently work for dynamically assembling services at run time in a massively distributed environment.

Keywords: *Web services; Autonomous Matchmaking; Semantic Web; RDF; Linked Open Data*

I. INTRODUCTION

Dynamically assembling services at run-time for developing massively distributed and interoperable systems [1] is an ultimate goal of Web services. Using XML via HTTP as the communication standard to exchange data between client applications and remote functionalities is the current standard of Web services, which is built around WSDL, SOAP and UDDI for completing the lifecycle of service description, publication and invocation. In the past decade, many research efforts have been made to realize the ultimate goal by adding value to the current standards. However, most of today's Web service applications are still developed in static and RPC/Document style [2].

These standards only represent the functional data structure and the syntax of a service [3], which ask service requesters to do most of the work manually. As a result, the automation level of communications among service requesters, broker and services is low. For example, clients find it difficult to automatically invoke services at run time because they need to manually build invocation SOAP messages based on the parameter orders described in the WSDL file although the invocation skeleton may be generated on the fly. Moreover, clients require prerequisite knowledge of each parameter's meaning by reading the service release document in order to correctly assign the parameters. Communication between broker and service requesters is even worse as no service

request protocol has been defined as yet, which makes dynamic service discovery impossible. Furthermore, UDDI has nearly disappeared from the literature defined in Web service lifecycle and most application developers directly use Web services based on their own knowledge. In order to solve these issues, Semantic Web technologies have been deployed to equip Web services. However, can Semantic Web Service (SWS) technology alone solve the dynamic problem?

The most recent SWS technologies can be divided into two different processes: (1) top-down process is defined by using domain ontologies, such as WSMO [4] and OWL-S [5]; (2) bottom-up process uses light-weight service annotations, such as WSMO-lite [7] and SAWSDL [3]. Both processes just move the hard discovery work from requester's side to the broker's side. In SWS environments, services need to publish either semantic description files or annotations into brokers in order to be discovered and invoked by requesters. Thus, brokers have to take a very heavy workload acting as a central point.

In spite of all these research efforts, the automation level has not dramatically increased. One main reason is the dissevered description layers of syntax and semantics. Syntactic descriptions such as WSDL and SOAP are still important for service invocation. Meanwhile, semantic descriptions or annotations only represent the syntax with semantics but they are nothing to do with services themselves to affect service behavior and invocation. In other words, current SWS approaches merely focus on enriching semantics for syntax without considering the actual data structure definitions that are very important for applications at run-time. Thus, semantic brokers can facilitate automatic service discovery, but run-time service invocation is still a big issue to prevent achieving the initial goal of Web services.

When the idea of Web services was born, the Semantic Web concept was not there yet. Why can we not go back to see whether we could re-think about Web services standards from the perspective of Semantic Web at the start? Most recent development of Linked Open Data (LOD) [6] gives us a new opportunity to link services together and specify services in a global unified semantics. In this paper, we view Web services with semantics from a different angle and introduce Autonomous Matchmaking Web Services (AMWS) based on RDF and Linked Open Data. In AMWS, all the communication protocols in the lifecycle are RDF messages. Most importantly, Web services, requesters and registry share equal workload,

which makes dynamically discovering, assembling and invoking more efficient and realistic to be achievable.

The following summarizes the roles of Web services, requesters and registry in AMWS:

- The requester needs to semantically describe the desired requirements about the requested Web services and send these requirements to the registry.
- The registry needs to pass the semantic requirements to all Web services that are registered with it and select or orchestrate services based on Web services' semantic responses about whether they are qualified to the requirements.
- Web services need to be aware the semantic requirements to notify the registry whether they satisfy the requirements.

The key contribution of this paper is to start use Semantic Web technologies throughout the whole Web services development, brokerage and consumption lifecycle and all three parts of Web services, service requester and service broker are semantic-aware.

The remainder of this paper is organized into three sections. Section II discusses the background and related work. Section III introduces the motivations. Section IV explains the Autonomous Matchmaking Web service framework in all details. Section V finally draws the conclusion and outlines the future work.

II. BACKGROUND AND RELATED WORK

A. Big Web Services vs. RestFul Services

W3C defines Web services¹ as "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically Web Services Description Language WSDL). Other systems interact with the Web services in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards." The Web services implemented in this definition are usually called Big Web services. Critics argue that Big Web services are too complex and based upon large software vendors or integrators, rather than typical open source implementation. Moreover, with an XML-based language it is difficult to identify the right construct to express a data model in a way that is fully supported by all SOAP/WSDL implementations [10].

With the popularity of Web 2.0, software functionalities accessible via HTTP (i.e. "Web services") are becoming the main underlying feature, which facilitates easy data exchange across the Web. Therefore, in contrast to Big Web Services, RestFul services implemented by using the PUT, GET and DELETE HTTP methods alongside POST become more popular. RestFul services are often better integrated with HTTP and web browsers than SOAP-based services. They do not require XML messages or WSDL-like service definitions.

However, the major limitation of RestFul services is lacking of basic standards to support service discovery and dynamic output parsing.

B. Light-weight Service Annotations and LOD

The main conceptual frameworks and specifications for semantically describing services (e.g. WSMO, OWL-S and SAWSDL which derive from WSDL-S [11]) are very comprehensive. Most SWS initiatives were built upon the enrichment of WSDL Web services with semantics. Moreover, these comprehensive semantic standards are too heavy to show the usability to the industry. It is only most recently that light-weight services (e.g. Web APIs and RESTful services) and service annotations have been researched. The main results of these recent studies are SA-REST [11], WSMO-Lite and MicroWSMO [12]. However, these changes are still focusing on service annotations for implementing a big middle broker layer rather than thinking of adding semantic values inside services.

Over the last few years, a significant portion of research on the Semantic Web has been devoted to create what is referred as LOD. LOD is a way to publish data on the Web in order for machines to understand the explicit meaning of the data. The data is linked to other external data sets, and can in turn be linked from external data sets. Meanwhile, LOD is based upon a set of principles, including the usage of HTTP URIs to provide information and allowing access based on RDF and SPARQL. Since these principles were outlined, there has been a large uptake, most notably through DBpedia² to produce a vast amount of linked datasets on the Web.

With the potential of LOD, service-oriented architecture can use the dataset directly to develop semantic services rather than to add semantic value later. In fact, LOD has been proposed as an approach for publishing and describing services, namely linked services [13] and Linked Open Services³. As a result, the service annotations are part of the LOD cloud.

C. Context-aware Web services

Service's performance adapting to dynamic changes influenced by meaningful inputs is a new Web services movement introduced in [14] and [15]. The basic principle is to enable services to understand the context of a service request, (e.g. input parameters and non-functional properties) and to provide the corresponded results. However, this process is only suitable for a limited scale of applications because the context-aware ontology is only specified at the domain level. Moreover, it is very unrealistic to match all possible performance to all possible contexts in one service and specific domain, excepting a manually negotiate process is required before the service invocation. For example, the different inputs will affect the speed of the service responding. However, the idea of Context-aware Web services gives an illumination of meaningful inputs can enhance the understandability between services and requesters at run-time.

¹ http://en.wikipedia.org/wiki/Web_service

² <http://dbpedia.org/About>

³ <http://www.linkedopenservices.org/>

III. MOTIVATION

In this section, we give two scenarios that have two basic requirements of dynamic service discovery and runtime service invocations.

a) *Context-aware applications in a ubiquitous environment.*

Context is defined as “meta-information to characterize the specific situation of an entity, to describe a group of conceptual entities, and to partition a knowledge base into manageable sets or as a logical construct to facilitate reasoning services” [8].

Based on this context definition, we introduced a typical context-aware application scenario [9] for Personalized Semantic News in the EU-funded NoTube project⁴ as follows:

A NoTube platform user acquires news items from generic broadcast streams and obtains additional enriched news information by using a set of personalized news related services. The platform should enable the use of user profile information and preferences to match the available news services. For example a user demands interesting news when he/she is using an iPhone and travelling by bus. His/Her profile describes that he/she prefers to use English and is generally interested in sports. The application should enable the user to get the interesting news data by discovering, selecting and invoking the suitable news services that match the user's context.

b) *E-Learning applications for learning content sharing and exchanging.*

In most e-Learning applications, sharing and exchanging learning objects in a multiplicity of distributed environment are the important requirement. In the EU-funded mEducator project⁵, there is a scenario about searching, publishing and creating learning contents for different topics and languages from/to multiple and different medical Learning Object Management Services (LOMS). In the meantime, each LOMS has its own input and output specifications. Moreover, the LOMSs can be added into the environment at any time when more education institutes joined. The application should enable dynamically invoking the suitable services to perform the functions.

IV. AUTONOMOUS MATCHMAKING WEB SERVICE FRAMEWORK

A. The Principles

The implementation and consumption of Autonomous Matchmaking Web services must follow four basic principles and the overall run-time lifecycle is represented in Figure 1.

1) *One service includes two layers, namely the autonomous matchmaking layer and the functionality layer, and two invocation endpoints for each layer respectively.* The autonomous matchmaking layer receives service searching message (SSM) from the registry and sends back “yes” or “no” confirmation response message (CRM) to the SSM

sender. The functionality layer receives service invocation input message (SIIM) and sends back a matched output message (MOM) which was defined inside the previous SSM.

2) *The service registers a service semantic annotations (SSA) as RDF into service registry and has the ability of identifying the function capability.* The SSA includes at least the ground information about the two invocation endpoints and non-functional properties, such as response time, license type and fees. Since the service itself will identify the function capability when receiving SSM, then publishing the functional semantic is not necessary.

3) *The service registry is able to identify the right service(s) and send back the Invocation Endpoint Message (IEM) to the service requester.*

4) *All messages are RDF with semantic annotations on each entity and the semantics are referenced by LOD.* For example, a FOAF ID defined in LOD Cloud can be used to annotate a userId entity which is one parameter of a input message (a clearer example will be illustrated later).

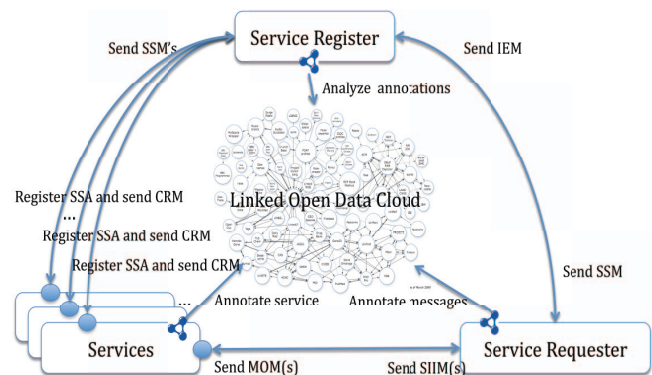


Figure 1. Run-time lifecycle of Semantic-aware Web services

B. Message Definitions

- Service Searching Message (SSM)

SSM is designed to specify the requirement of the desired service(s) from the service requester's point of view. The ultimate goal of SSM is to allow the service autonomous matchmaking layer to understand what the requester needs. There are two major advantages: (1) SSM is a message (not service annotation) protocol that is purely defined by the needs of application developments at design time and is searching the desired service at run time when communicating to services through Registry via the message. (2) SSM aims to use global understandable semantic references of LOD, although a domain specific ontology is also allowed. In this way, the service autonomous matchmaking layer can decide whether the service functionality is suitable according to the SSM. The first-draft RDF schema of SSM is defined in Figure 2.

Each SSM includes at least functional requirements of the desired service and the brokerage mode attribute. The specification of non-functional requirements is an

⁴ <http://www.notube.tv/>

⁵ <http://www.meducator.net/>

optional part to enhance the brokerage process for selecting service(s).

The *hasMode* property is an enum data type defining two elements: “single” and “set”. The “single” indicates only one best suitable service is requested and the “set” means that all suitable services are required. Because *hasMode* is only useful for the registry, it will not pass to Web services and SSM’s (in Figure 1.) are the SSM messages without *hasMode* property.

The *FunctionalRequirement* class consists of *InputMessage*, *OutputMessage* and *ServiceCategory*. *InputMessage* and *OutputMessage* include *Parameters* what are composed by one *Element* or more. *ServiceCategory* indicates service domain. The most important part of the SSM schema is to use global recognizable RDF entities to semantically reference the *Element* and *ServiceCategory*. Based on current semantic web standards, LOD is most suitable resource to be applied. For example, the Service Finder RDFs⁶ can be one of the *ServiceCategory* references.

The *NonFunctionalRequirement* class includes nonfunctional parameters that can be semantically referenced to specify the properties like response-time, fee and language.

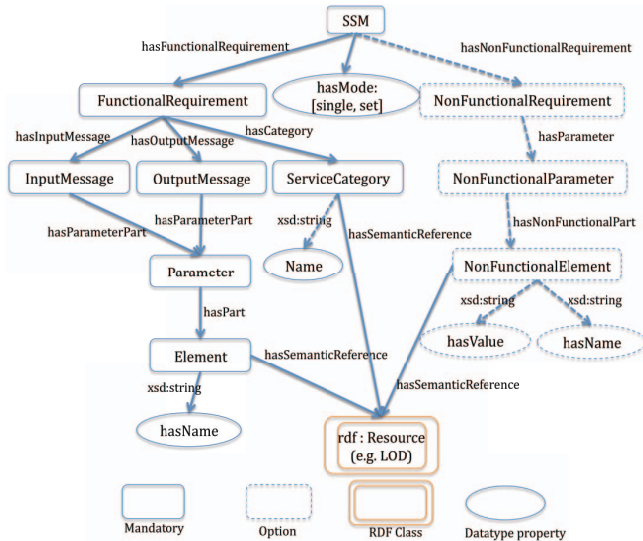


Figure 2. First-draft of SSM RDF schema

- Confirmation Response Message (CRM)

CRM is a simple message to confirm whether the service is suitable by sending to the SSM sender. The first-draft RDF schema of CRM is defined in Figure 3.

The *hasRegistrationID* property is a unique identifier that is registered and links to other service information in the service registry, for instance, non-functional properties and request endpoint.

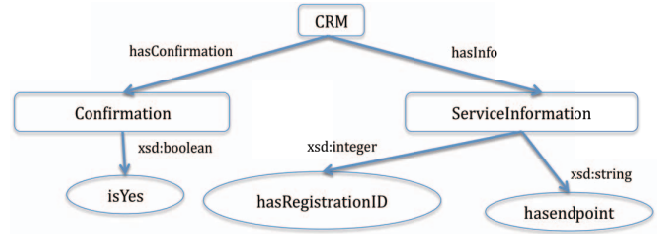


Figure 3. First-draft of CRM RDF schema

- Invocation Endpoint Message (IEM)

An instant message of IEM is sent from the service registry to service requester for supporting the invocation endpoint(s). Based on the service requested *hasMode* property defined in SSM, the registry will decide whether a set of service endpoints or single service endpoint should be included in the message. The first-draft of the IEM RDF schema shows in Figure 4.

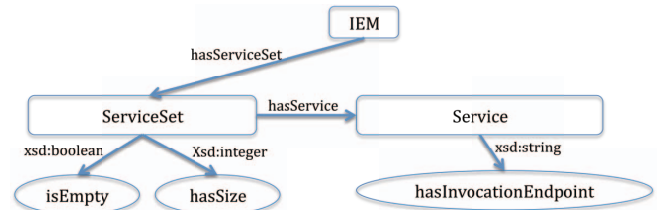


Figure 4. First-draft of IEM RDF schema

- Service Invocation Input Message (SIIM)

When the service requester gets the invocation endpoint(s), (an) instant SIIM(s) will be sent to these endpoint(s) for service invocation. The first-draft of SIIM RDF schema is illustrated in Figure 5.

As defined in SSM, the *Element* included in *Parameter* of *InputMessage* is semantically referenced to enable service side to correctly retrieve the input data.

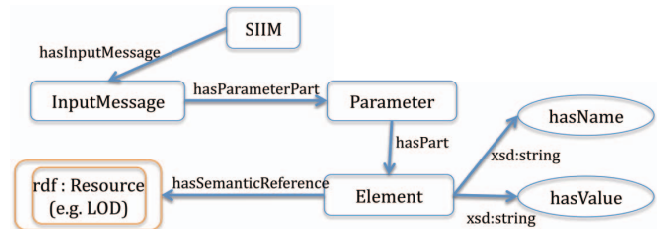


Figure 5. First-draft of SIIM RDF schema

- Matched Output Message (MOM)

All response messages from invoked services follow MOM RDF schema. MOM is very similar to SIIM but change the *Element* input value to the *Element* output value as displayed in Figure 6. This time, the semantics of *Element* is used by the service requester to finally pickup the correct response data.

⁶ <http://www.service-finder.eu/ontologies/ServiceCategories>

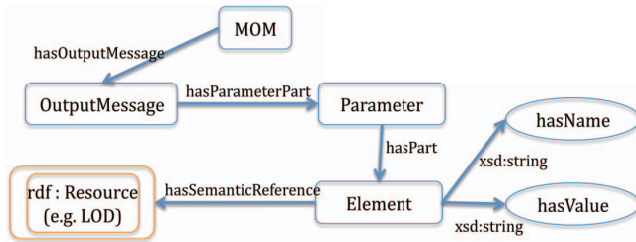


Figure 6. First-draft of MOM RDF schema

C. Benefits

There are two major benefits of applying the AMWS framework.

1) All information and communication messages are semantically understandable by using unified RDF data structure and LOD semantics. As result, all three parts can know the data structure and semantics at the same time, which is a fundamental requirement to enable services to be dynamically assembled and invoked.

2) The workload among Web services, Service registry and service requester to achieve dynamically assembling and invoking services are trade-off. Each part of the three takes their own responsibilities to efficiently finish the service consumption life-cycle. Therefore, AMWS framework is suitable for large scale distributed applications.

D. Service development suggestions for the scenarios

To implement the AMWS in both context-aware and e-Learning scenarios requires four steps:

Step 1. Describing and storing service properties with semantics.

For example, the news service from the context-aware scenario takes topic and keywords as input parameters and produces title description and stream URIs as output parameters. The service providers should have their own service specification to enable comparing it to the SSM. The document in Figure 7 shows an example of storing the input message specification as RDF.

The *hasSemanticReference* properties being highlighted is the key elements in the document. In the similar way, the output message can be specified as a RDF document as well. When receiving SSM, the service first responds to the registry whether it is suitable. When the service is invoked, it retrieves the semantic matched input parameters to produce the semantic matched outputs.

```
<rdf:RDF>
  dfs:Class df:about="http://open.ac.uk/semanticWS/InputMessage"/I
  dfs:Class df:about="http://open.ac.uk/semanticWS/SPa amete "/I
  dfs:Class df:about="http://open.ac.uk/semanticWS/Element"/I
  dfs:ObjectP ope ty df:j boyt="http://open.j c.yk/semj nticWS/hj sPj t"l
    dfs: j nge df: esoy ce="http://open.j c.yk/semj nticWS/Element"/I
    dfs:domj in df: esoy ce="http://open.j c.yk/semj nticWS/Pj j mete "/I
  / dfs:ObjectP ope tyl
  dfs:ObjectP ope ty
  df:j boyt="http://open.j c.yk/semj nticWS/hj sPj j mete Pj t"l ...
  dfs:Dj tj typeP ope ty
  df:j boyt="http://open.j c.yk/semj nticWS/hj sNj me"l ...
  dfs:Dj tj typeP ope ty
  df:j boyt="http://open.j c.yk/semj nticWS/hj sSemj ntics efence"l
  dfs:domj in df: esoy ce="http://open.j c.yk/semj nticWS/Element"/I
  dfs: j nge df: esoy ce=".../_MLSchemj #st ing"/I
  / dfs:Dj tj typeP ope tyl
  Element df:j boyt="...#InpytMessj ge_veywo ds_element"l
  hj s#j me df:dj tj type=".../_MLSchemj #st ing"l
  veywo ds /hj s#j mel
  hj sSemj ntics efence df:dj tj type=".../_MLSchemj #st ing"l
  hj sSemj ntics efence df:dj tj type=".../_MLSchemj #st ing"l
http://www.tj lwdigge .com/conve sj tions/web.mit.edy/newsoffice/veywo ds
  /hj sSemj ntics efence l
  /Element l
  Pj j mete df:j boyt="...#InpytMessj ge_veywo ds"/I
  InpytMessj ge df:j boyt="...#InpytMessj ge_news"l
  hj sPj j mete Pj tl
  Pj j mete df:j boyt="...#InpytMessj ge_topic"l
  hj sPj tl
  Element df:j boyt="...#InpytMessj ge_topic_element"l
  hj sSemj ntics efence df:dj tj type=".../_MLSchemj #st ing"l
http://www.tj lwdigge .com/conve sj tions/web.mit.edy/newsoffice/topic
  /hj sSemj ntics efence l
  hj s#j me df:dj tj type=".../_MLSchemj #st ing"
  l topic /j.0:hj s#j mel
  ...
```

Figure 7. An example of a RDF document provided by service providers for describing and storing service properties with semantics

1) Step 2. Implementing services.

Services should be implemented according to the described service properties (in our case, the RDF descriptions) and grounded with an invocation endpoint.

2) Step 3. Developing SSM comparing mechanism with a Autonomous Matchmaking endpoint.

The comparing mechanism should define the rules of acceptable SSMs. For example, if the $\text{input_service} \supseteq \text{input_requirement}$ and $\text{output_service} \subseteq \text{output_requirement}$, then the SSM is acceptable and the service will send a “yes” response to the registry. Otherwise, a “no” response is sent. If the SSM includes non-functional properties, then the non-functional property comparing mechanism should be defined or leave it to the registry to decide.

3) Step 4. Publishing endpoints to the registry.

The two endpoints of Autonomous Matchmaking and invocation should be published into the registry. The non-functional properties are optional to be published based on whether services desire to be brokered.

V. CONCLUSION AND FURTHER DISCUSSIONS

In this paper we introduced a new Web services framework namely AMWS: Autonomous Matchmaking Web services. The AMWS is based on the most recent Semantic Web and Web services research results aiming to achieve dynamic service discovery, assembling and invocation in a large-scale, distributed environment. The main ideas are (1) the AMWS uses RDF messages as a communication protocol among services, requesters and the registry; (2) the RDF entities are referenced by LOD dataset for giving the semantics and for filling the knowledge gap between requesters and services; (3) the AMWS uses Autonomous Matchmaking to notify the suitability to the registry, which better fits into the distributed environment than typical WS standards and SWS frameworks.

The AMWS is a first attempt to refine the WS or SWS discovery, assembling and invocation lifecycle by just using Semantic Web technology to develop services rather than adding semantic layers to the syntax based WS blocks. However, the AMWS approach is still at the very early stage and it has many open questions that need to be answered. For instance, is autonomous matchmaking necessary when a broker is there? One answer could be “yes”, because it distributes the discovery workloads from the centralized broker. Moreover, Autonomous matchmaking can reduce the fault rates at runtime if a service changes its behavior or takes different service requirements to modify its own behavior like context-aware services. The other answer could be “no”, if the centralized broker is allocated in a powerful machine or has powerful distributed calculation mechanism such as Grid computing and services are very stable. The other issue may be related to using RDF not OWL or other semantic standards. We have to say that this is just based on current industry practice on RESTful Web services that produce mainly RDF results and one reason could be RDF is easier to be grounded than OWL and other standards.

This paper aims to start to reconsider Web services using Semantic Web eyes in order to resolve current Web services and SWS problems when dynamically discovering, assembling and invoking services. Our future work will involve industry partners to investigate the Autonomous Matchmaking mechanism, usability and practicability to improve the AMWS

framework. Furthermore, a more comprehensive Autonomous Matchmaking mechanism will be studied.

ACKNOWLEDGMENT

The work is supported in part by the European Commission under Grant ECP2008EDU418006 for mEducator project and FP7-ICT-231761 for NoTube project.

REFERENCES

- [1] Papazoglou, M. P., Traverso, P., Dustdar, S. and Leymann, F., Service-Oriented Computing: A Research Roadmap, *International Journal of Cooperative Information Systems*, Vol. 17, No. 02. (2008), 223.
- [2] Gunzer, H and Engineer, S., Introduction to Web Services, Borland Developer Network (2002), DOI=<http://bdn.borland.com/article/images/28818/webservices.pdf>.
- [3] Kopecký, J., Vitvar, T., Bournez, C. and Farrell, J., "SAWSDL: Semantic Annotations for WSDL and XML Schema," *IEEE Internet Computing*, vol. 11, no. 6, pp. 60-67, Nov./Dec. 2007, doi:10.1109/MIC.2007.134
- [4] WSMO Working Group (2004), D2v1.0: Web service Modeling Ontology (WSMO). WSMO Working Draft, (2004). (<http://www.wsmo.org/2004/d2/v1.0/>).
- [5] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K. (2004). OWL-S: Semantic Markup for Web Services. Member submission, W3C. W3C Member Submission 22 November 2004.
- [6] Bizer, C., Heath, T., Berners-Lee, T., Linked data - The Story So Far, Special Issue on Linked data, *International Journal on Semantic Web and Information Systems (IJSWIS)*, 2009.
- [7] WSMO-lite, DOI= <http://cms-wg.sti2.org/TR/d11/v0.2/>.
- [8] Boukadi, K., et al. CWSC4EC: How to Employ Context, Web Service, and Community in Enterprise Collaboration. *NOTERE '08: Proceedings of the 8th int conference on New technologies in distributed systems*, 2008.
- [9] Yu, H. Q., Benn, N., Dietze, S., Siebes, R., Pedrinaci, C., Liu, D., Lambert, D., and Domingue, J., (2010) Two-staged approach for semantically annotating and brokering TV-related services, *The IEEE International Conference on Web Services (ICWS)*, Miami, Florida.
- [10] Pautasso, C., Zimmermann, O., and Leymann, F., 2008. Restful web services vs. "big" web services: making the right architectural decision. In *Proceeding of the 17th international Conference on World Wide Web* (Beijing, China, April 21 - 25, 2008). WWW '08. ACM, New York, NY, 805-814. DOI= <http://doi.acm.org/10.1145/1367497.1367606>.
- [11] Sheth, A. P., Gomadam, K., and Ranabahu, A., 2008. Semantics enhanced services: Meteor-s, SAWSDL and SA-REST. *IEEE Data Eng. Bul l.*, 31(3):8-12.
- [12] Maleshkova, M., Pedrinaci, C., and Domingue, J. 2009. Supporting the creation of semantic restful service descriptions. In *Workshop: Service Matchmaking and Resource Retrieval in the Semantic Web (SMR2) at 8th International Semantic Web Conference*.
- [13] Pedrinaci, C., Domingue, J., and Krummenacher, R. 2010 Services and the Web of Data: An Unexploited Symbiosis, *Workshop: Linked AI: AAAI Spring Symposium "Linked data Meets Artificial Intelligence"*.
- [14] Yu, H. Q., Reiff-Marganiec, S., "A Method for Automated Web Service Selection", services, pp. 513-520, 2008 *IEEE Congress on Services - Part I*, 2008.
- [15] Truong, H. and Dustdar, S., A Survey on Context-aware Web Service Systems, *International Journal of Web Information Systems*, 5(1):5 - 31, (c) Emerald, 2009.